DETAILED ACTION

Remarks

1. This office action is in response to the amendment filed on 8/05/2009.
2. Claims 1, 11 and 43-46 have been amended.
3. Terminal Disclaimer filed on 11/19/2007 no longer applied, as the conflict co-pending application 10/377,362 has already been abandoned on 09/16/2009.
4. Claims 1-4, 6, 7, 9, 11-14, 16, 17, 19, 31-34 and 43-46 remain pending and have been examined.

EXAMINER'S AMENDMENT

5. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.
6. Authorization for this examiner's amendment was given in a telephone interview and further email communication with Karl Kenna(Reg# 45,445) on 12/03/2009 to incorporate limitations disclosed in specification and to put the claims in condition for allowance.
7. Applicant's proposed amendment is received via email and is accepted by examiner.
8. Claims 1, 11 and 45 are now being further amended based on the received applicant's proposed amendment.

9. The application has been amended as follows:

IN THE CLAIMS

Please amend claims 1, 11 and 45 as follows:

1. (Currently Amended) A system for loading software applications, comprising:

a server, executing a Java virtual machine that includes a system classloader, for storing and running a plurality of software applications, wherein each of said plurality of software applications includes a plurality of deployable modules and classes associated therewith, and wherein the software applications or modules therein can be customized by a software developer and then deployed to run on the same server;

a control file, that can be edited by the software developer and associated with said plurality of software applications or modules, wherein said control file specifies a hierarchy of application classloaders as children of the system classloader, to be used with the modules in said plurality of software applications, and wherein the hierarchy includes a plurality of nested branches and siblings, including providing each of said plurality of software applications or modules with its own application classloader hierarchy so that the software applications or modules are not aware of classloaders or classes that are assigned to another software application, and wherein the hierarchy is specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications and wherein the system classloader includes only application level library classes, and wherein the nested branches of the hierarchy includes classes or modules that are frequently called by modules in said plurality of software applications as siblings within the same branch or same classloader; and

a deployment utility that, upon receiving a request to deploy and run a software application on the server,

parses the control file and determines which classloaders are specified therein for the software application being deployed,

loads with said software application into the Java virtual machine at the server a selection of said application classloaders corresponding to the hierarchy specified by said control file, including creating sibling classloaders for implementations whose classes are to be loaded separately as specified by the control file, and, if a particular software application or a module in a software application is being redeployed then

loading only the application classloaders that are specified in the branches for that particular software application or module, without loading any of the other branches in the hierarchy, and

enables the server to host multiple isolated software applications or modules within the Java virtual machine, as defined by the hierarchy;

wherein, upon receiving a call to instantiate an implementation, the system searches within the hierarchy including first searching the calling classloader, and then successively searching parent classloaders above the calling classloader, to locate and instantiate necessary classes or objects.

11. (Currently Amended) A method for loading software applications on a server executing a Java virtual machine that includes a system classloader, comprising the steps of:

providing a server executing a Java virtual machine for storing and running a plurality of software applications;

providing a plurality of software applications, wherein each of said plurality of software applications includes a plurality of deployable modules and classes associated therewith, and wherein the software applications or modules therein can be customized by a software developer and then deployed to run on the same server;

providing a control file associated with said software application or modules, wherein said control file can be edited by a software developer and specifies a hierarchy of application classloaders as children of the system classloader, to be used with the modules in said software application, and wherein the hierarchy includes a plurality of nested branches and siblings, including providing each of said plurality of software applications or modules with its own application classloader hierarchy so that the software applications or modules are not aware of classloaders or classes that are assigned to another software application, and wherein the hierarchy is specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications, and wherein said application classloaders are children of system classloader and wherein the system classloader includes only application level library classes, and wherein the nested branches of the hierarchy includes classes or modules that are frequently called by modules in said plurality of software applications as siblings within the same branch or same classloader; and

upon receiving a request to deploy and run a software application on the server,

parsing the control file and determining which classloaders are specified therein for the software application being deployed,

retrieving a selection of said application classloaders according to the hierarchy specified by said control file, and

loading said modules and classes into the Java virtual machine at the server as part of said software application corresponding to said hierarchy, including creating sibling classloaders for implementations whose classes are to be loaded separately as specified by the control file, and, if a particular software application or a module in a software application is being redeployed then loading only the application classloaders that are specified in the branches for that particular software application or module, without loading any of the other branches in the hierarchy, to enable the server to host multiple isolated software applications or modules within the Java virtual machine, as defined by the hierarchy;

wherein, upon receiving a call to instantiate an implementation, the system searches within the hierarchy including first searching the calling classloader, and then successively searching parent classloaders above the calling classloader, to locate and instantiate necessary classes or objects.

45. (Currently Amended) A computer readable storage medium, including instructions stored thereon which when read and executed by a computer cause the computer to perform the steps comprising:

providing a server executing a Java virtual machine for storing and running a plurality of software applications;

providing a plurality of software applications, wherein each of said plurality of software applications includes a plurality of deployable modules and classes associated therewith, and wherein the software applications or modules therein can be customized by a software developer and then deployed to run on the same server;

providing a control file associated with said software application or modules, wherein said control file can be edited by a software developer and specifies a hierarchy of application classloaders as children of the system classloader, to be used with the modules in said software application, and wherein the hierarchy includes a plurality of nested branches and siblings, including providing each of said plurality of software applications or modules with its own application classloader hierarchy so that the software applications or modules are not aware of classloaders or classes that are assigned to another software application, and wherein the

hierarchy is specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications, and wherein said application classloaders are children of system classloader and wherein the system classloader includes only application level library classes, and wherein the nested branches of the hierarchy includes classes or modules that are frequently called by modules in said plurality of software applications as siblings within the same branch or same classloader; and

upon receiving a request to deploy and run a software application on the server,

parsing the control file and determining which classloaders are specified therein for the software application being deployed,

retrieving a selection of said application classloaders according to the hierarchy specified by said control file, and

loading said modules and classes into the Java virtual machine at the server as part of said software application corresponding to said hierarchy, including creating sibling classloaders for implementations whose classes are to be loaded separately as specified by the control file, and, if a particular software application or a module in a software application is being redeployed then loading only the application classloaders that are specified in the branches for that particular software application or module, without loading any of the other branches in the hierarchy, to enable the server to host multiple isolated software applications or modules within the Java virtual machine, as defined by the hierarchy;

wherein, upon receiving a call to instantiate an implementation, the system searches within the hierarchy including first searching the calling classloader, and then successively searching parent classloaders above the calling classloader, to locate and instantiate necessary classes or objects.

--END OF AMENDMENT--

Allowable Subject Matter

10. **Claims 1-4, 6, 7, 9, 11-14, 16, 17, 19, 31-34 and 43-46 are allowed** (re-numbered as claims 1-22). As the Applicants pointed out under REMAKRS section, page number 9-14, neither Taylor, Susarla, Peterson, alone or in combination, teach or suggest the claimed invention as claimed in independent claims 1, 11 and 45. Specifically, the combination of Taylor, Susarla and Peterson fails to teach or suggest the feature about "a control file, that can be edited by the software developer and associated with said plurality of software applications or modules, wherein said control file specifies a hierarchy of application classloaders as children of the system classloader, to be used with the modules in said plurality of software applications, and wherein the hierarchy includes a plurality of nested branches and siblings, including providing each of said plurality of software applications or modules with its own application classloader hierarchy so that the software applications or modules are not aware of classloaders or classes that are assigned to another software application, and wherein the hierarchy is specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications, wherein the system classloader includes only application level library classes, wherein the nested branches include specifying classes or modules located in siblings that are frequently called by the modules in said plurality of software applications or modules in the same branch or in the same classloader" as further clarified by the Examiner's Amendment above, and

in as such manners as recited in the independent claims 1, 11 and 15. Thus Claims 1, 11, 15 and each of the dependent claims are allowable for at least the same reasons.

11. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Zheng Wei whose telephone number is (571) 270-1059 and Fax number is (571) 270-02059. The examiner can normally be reached on Monday-Thursday 8:00-15:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic

Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Z. W./
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192